**Web安全测试指南**

**文档修改记录**

|  |  |  |
| --- | --- | --- |
| 批准日期 | 修订说明 | 编制部门 |
| 2016/2/4 | 评审定稿 | EPG |
| 2016/02/26 | 正式发布 | EPG |

目录

[编写目的 4](#_Toc443635219)

[一、敏感数据 5](#_Toc443635220)

[1．传输 5](#_Toc443635221)

[2．后台信息 5](#_Toc443635222)

[二、验证码检查（短信&邮件&图形） 9](#_Toc443635223)

[三、弱密码管理 10](#_Toc443635224)

[四、权限管理 11](#_Toc443635225)

[1．越权操作 11](#_Toc443635226)

[2．平行权限 11](#_Toc443635227)

[3．获取跨权限服务 12](#_Toc443635228)

[五、XSS攻击 12](#_Toc443635229)

[六、CSRF攻击 13](#_Toc443635230)

[七、Sql注入 13](#_Toc443635231)

[八、文件上传 16](#_Toc443635232)

[九、Web会话管理（session） 18](#_Toc443635233)

[十、Web服务器信息 20](#_Toc443635234)

# 编写目的

让普通测试人员通过对文档中内容的学习，建立起基本的安全测试思维和分析能力，了解相应的安全规范和测试范围，并掌握一些常用的安全测试方法及工具的操作技能，将其运用到实际测试工作中。

# 一、敏感数据

## 1．传输

明文传输，从广义的讲，指计算机于计算机之间进行数据传输时的方式，是一段未经过任何加密的数据。

在web应用和网站中，数据的传输多以http协议下的GET, POST等方法通过浏览器以请求/响应的方式进行通信。其中的敏感类数据，如账户密码、身份证号、银行卡号等，如果在未加密的情况加进行了传输，就极容易被中间的代理服务器或抓包工具所截获并篡改，其中的危害不言而喻。例如在登陆、资金交易等操作流程中，特别要注意是否存在敏感信息的明文泄露。对于登陆操作，更要求使用HTTPS协议来保障传输安全。

抓包工具有哪些？百度一搜一大把，如burpsuite、webscarab、fiddler

|  |  |
| --- | --- |
| 测试用例名称 | 传输过程中机密性信息保护 |
| 测试目的 | 测试Web程序在处理账户密码、身份证号、银行卡号等传输是否采用了加密传输的机制。 |
| 测试条件 | 1. 已知Web网站地址 2. Web业务正常运行 |
| 执行步骤 | 1. 已知网站登录地址为：http://www.exmaple.com/login.xxx 2. 开启WebScarab，配置对GET和POST请求进行拦截；在浏览器中配置代理服务器IP为127.0.0.1，端口为8008 3. 在登录处输入用户名和口令、验证码登陆 4. 查看WebScarab拦截的请求中，用户名和口令是否已经加密；且是否采用HTTPS协议传输。   **备注：其他业务请求操作过程与以上步骤一致** |
| 预期结果 | 用户机密信息采用HTTPS传输。 |
| 测试工具 | burpsuite、webscarab、fiddler |

## 2．后台信息

1）用户密码属于机密信息，需要在数据库加密存储，要求不可逆、不可暴力破解，因此需要检查数据库中密码是否加密。

|  |  |
| --- | --- |
| 测试用例名称 | 用户机密信息加密存储 |
| 测试目的 | 用户机密信息，需要在数据库加密存储，要求不可逆、不可暴力破解。 |
| 测试条件 | 1. 已知数据库地址 2. 用户机密信息所在字段（项目组提供） 3. 有数据库访问权限 |
| 执行步骤 | 1. 数据库客户端工具登上数据库 2. 查看用户机密信息所在字段 |
| 预期结果 | 用户机密信息字段在数据库中均已经加密存储；  不同的用户使用同个密码，密文不一样。（建议） |
| 测试工具 | 访问数据库客户端工具 |

2）日志中不能包含敏感信息。此项检查通过源代码搜索、查询对应打印语句。

|  |  |
| --- | --- |
| 测试用例名称 | 日志中不能包含敏感信息 |
| 测试目的 | 用户机密信息（哪些是机密信息项目组提供），不能在日志中打印 。 |
| 测试条件 | 1. 有登录web服务器查看权限 2. 调整日志级别到debug级别 3. 源代码查看权限 |
| 执行步骤 | 1：查询打印日志，搜索机密信息关键字（关键字由项目组提供）。  2：在源代码中搜索打印机密信息关键字 |
| 预期结果 | 1：用户机密信息都加密输出  2：源代码中不存在打印机密信息语句，如下图所示：  C:\Users\hezf\AppData\Roaming\Foxmail7\Temp-4408-20151028081157\_CXB7]5YNQB1KRW{(10-28-10-31-32).png |
| 测试工具 | 无 |

3）错误页面检查，不可以泄漏源码。不允许将代码堆栈打印在页面上。

|  |  |
| --- | --- |
| 测试用例名称 | 不允许将错误代码堆栈打印在页面上 |
| 测试目的 | 错误代码堆栈中包含很多源代码信息，避免此类信息给黑客进一步攻击提供信息。 |
| 测试条件 | 具有能访问应用所有业务的各类用户权限。 |
| 执行步骤 | 1. 使用扫描工具 2. 执行业务测试过程 |
| 预期结果 | 没有错误代码堆栈打印在页面上；  所有错误页面信息必须有统一处理。 |
| 测试工具 | AppScan、AWVS、明鉴Web应用弱点扫描器 |

4）数据库连接配置密码需要加密。检查数据库连接配置文件，是否该密码为明文。

|  |  |
| --- | --- |
| 测试用例名称 | 数据库连接配置密码需要加密 |
| 测试目的 | 普通人员不能获得连接数据库的权限 |
| 测试条件 | 1. 已经知道连接数据库配置文件名及存放位置 2. 有源代码工程的读取权限 |
| 执行步骤 | 查看该文件 |
| 预期结果 | 数据库连接密码已经加密。  加密方式不能是base64，字符变化等容易被猜解的方式。 |
| 测试工具 | 无 |

5）缓存信息存储安全：不能将敏感数据泄漏到浏览器的缓存中。

参考：如何查看浏览过的本地缓存Cookie和网站数据，参考以下地址：

<http://jingyan.baidu.com/article/48b558e335b9ac7f38c09a8f.html>

(关于会话信息保护，详见第九章)

|  |  |
| --- | --- |
| 测试用例名称 | 浏览器缓存不存在用户机密信息 |
| 测试目的 | 用户机密信息不会在浏览器缓存中保存 |
| 测试条件 | 无 |
| 执行步骤 | 以chrome浏览器为例：   1. 查看cookie：打开浏览器查看工具。（如何查看见以上提示） 2. 查看非cookie的其他缓存：打开chrome://view-http-cache/，找到对应地址，检查该地址中是否存在用户机密信息以明文形式存在。 |
| 预期结果 | 1. 对应的业务应用不存在用户机密信息。 2. 对应的业务应用不存在用户机密信息以明文形式存在。（除cookie之外的其他信息需要再本地缓存中看） |
| 测试工具 | 无 |

6）检查工程配置文件的相关信息：包括管理员信息、内部IP信息、系统配置信息等不能在工程中存在。（可以通过扫描工具搜索，也可以手工对Web目录进行检查，或者检查工程配置文件。）

|  |  |
| --- | --- |
| 测试用例名称 | 检查工程中与应用无关的文件，例如：存在管理员信息、内部IP信息、应用系统配置信息等 |
| 测试目的 | 不能将管理员信息、内部IP信息、系统配置信息放在应用工程中 |
| 测试条件 | 1. 已知应用工程部署地址 2. 已有应用工程各类用户权限 |
| 执行步骤 | 1. 检查应用工程 2. 使用扫描工具扫描 |
| 预期结果 | 工程中不存在管理员信息、内部IP信息、系统配置信息等 |
| 测试工具 | AppScan、AWVS、明鉴Web应用弱点扫描器 |

# 二、验证码检查（短信&邮件&图形）

验证码在目前应用的系统中重要性越来越高，本小结分别对短信、邮件验证码与图形验证码的测试点进行了描述。检查的目的是为了防止恶意注册、爆破等攻击。

|  |  |
| --- | --- |
| 测试用例名称 | 短信和邮件验证码检查 |
| 测试目的 | 检查的目的是为了防止恶意注册、爆破等攻击 |
| 测试条件 | 有应用系统的普通用户权限 |
| 执行步骤 | 1. 使用新用户注册，不断往新用户手机号发送验证码。 2. 检查发送的验证码。 3. 检查验证码的失效时间。 4. 使用一次验证码后，再次使用该验证码。 5. 检查前台页面代码。 6. 往短信和邮件上发送验证码，检查应答请求。 7. 输入在有效期内的验证码，使用工具篡改对应的手机号或邮箱地址；发送请求。   **其他：**   * 建议一天内往同一个手机发送的次数在10次以上就被锁定一段时间。 * 建议在短信和邮件验证码之前需要有图片验证码。 |
| 预期结果 | 1. 限制在一定时间范围内对同一注册号发送验证码的次数，注册手机号被限时冻结。 2. 短信校验码长度要求6位或以上。 3. 验证码时效时间下10分钟范围内。 4. 成功了一次就失效；失败了短信、邮件可以试几次，但是必须是10次以内。 5. 不能找到验证码生成代码，验证码必须是服务端生成。 6. 应答中不能有验证码。 7. 请求失败。验证码与手机号或邮箱地址必须一一绑定。 |
| 测试工具 | BurpSuit、webscarab、 |

|  |  |
| --- | --- |
| 测试用例名称 | 图片验证码检查 |
| 测试目的 | 检查的目的是为了防止恶意注册、爆破等攻击 |
| 测试条件 | 有应用系统的普通用户权限 |
| 执行步骤 | 1. 检查发送的验证码。 2. 检查验证码的时效时间。 3. 检查图片验证码的表现形式。 4. 往手机上发送的验证码，检查应答请求。 5. 使用一次验证码后，再次使用该验证码。 6. 检查前台页面代码。 7. 输入在有效期内的验证码，使用工具篡改对应的手机号或邮箱地址；发送请求。 |
| 预期结果 | 1. 图形校验码长度要求6位或以上。 2. 图片验证码时效时间下2-5分钟范围内。 3. 图形数字、文字或图案、且有干扰。 4. 应答中不能有验证码。 5. 不能使用成功，验证码只能被使用一次；失败后会重新生成新验证码。 6. 不能找到验证码生成代码，验证码必须是服务端生成。 7. 请求失败；验证码与手机号或邮箱地址必须一一绑定。 |
| 测试工具 | BurpSuit、webscarab |

# 三、弱密码管理

|  |  |
| --- | --- |
| 测试用例名称 | 弱密码管理 |
| 测试目的 | 防止暴力破解 |
| 测试条件 | 有应用系统的访问权限，知道各处密码输入点。 |
| 执行步骤 | 检查密码输入 |
| 预期结果 | 1. 密码的最短长度（建议8位及以上、根据实际需求）。 2. 密码必须是大小写字母+数字+符号的组合。 3. 默认密码要求强制改密。 4. 如果多次登录失败、账户能被限时锁定。 5. 有周期性密码更新提示。（根据实际需求） |
| 测试工具 | 无 |

# 四、权限管理

## 1．越权操作

因为web应用中的操作步骤是与其访问地址相关联的，所以有时候一些乱序的请求页面操作，就会暴露出其中的安全问题。

对于此类漏洞的测试，可通过经常关注URL地址所包含的信息和参数来进行练习。

|  |  |
| --- | --- |
| 测试用例名称 | 越权操作（本用例请根据实际业务提炼用例，本用例仅举例） |
| 测试目的 | 防止通过越权操作，直接访问最终业务。 |
| 测试条件 | 一笔下订单的操作，页面的访问顺序为：  “选货->确认收货地址->确认订单信息->提交支付->确认支付->确认订单状态->发货” |
| 执行步骤 | 1. 操作完“确认订单信息”后直接访问“确认订单状态”或“发货”的URL地址。 |
| 预期结果 | 不能越权访问 |
| 测试工具 | 无 |

## 2．平行权限

直白的讲就是不同用户ID之间可以相互访问其信息，或有相关的操作权限。测试的要点除了像上面提的那样关注URL信息外，也可通过跟踪后台请求数据的方式排查。

|  |  |
| --- | --- |
| 测试用例名称 | 平行权限 |
| 测试目的 | 不同用户之间可以相互访问其信息，或有相关的操作权限。 |
| 测试条件 | A用户与B用户是同级用户 |
| 执行步骤 | 1. A用户登录。 2. A用户访问属于B用户权限的数据。 3. A用户操作属于B用户权限能操作的权限。 |
| 预期结果 | A用户不能访问B用户权限的数据、不能操作属于B用户权限能操作的业务。 |
| 测试工具 | BurpSuit、webscarab、 |

## 3．获取跨权限服务

通过普通用户账号去访问一些管理员账号才能访问的URL地址。

|  |  |
| --- | --- |
| 测试用例名称 | 获取跨权限服务 |
| 测试目的 | 普通用户不能访问一些管理员账号才能访问的URL地址 |
| 测试条件 | 无 |
| 执行步骤 | 1. 普通用户A登录。 2. A用户访问管理员admin才能访问的url |
| 预期结果 | A用户不能访问管理员admin才能访问的url |
| 测试工具 | 无 |

# 五、XSS攻击

跨站脚本攻击的手段主要是往Web页面里插入恶意的脚本代码，当用户浏览该页之时，嵌入其中Web里面的代码会被执行，从而达到恶意攻击用户的特殊目的。代码可能是html标签，也可能是JS语句等等。

在做这类测试的时候会遇到个问题，就是要测的web应用页面上可输入的编辑框太多，无法通过手工去完全覆盖，这时候可以先用爬虫工具扫描一下，把可疑的URL先过滤出来，再手工验证确认。

|  |  |
| --- | --- |
| 测试用例名称 | XSS攻击 |
| 测试目的 | 防止Web页面里插入恶意的脚本代码 |
| 测试条件 | 获得应用系统各种权限 |
| 执行步骤 | 1. 在输入框中输入<img src=1 onerror="alert(1)"> 或<script type="text/javascript">alert(XSS-test)</script> 2. 使用扫描工具进行扫描 |
| 预期结果 | 1. 表单内容提交后，页面不会弹出窗“1” 2. 工具未扫出可疑页面 |
| 测试工具 | AppScan、AWVS、明鉴Web应用弱点扫描器 |

# 六、CSRF攻击

CSRF（Cross-site request forgery跨站请求伪造，也被称为“One Click Attack”或Session Riding，通常缩写为CSRF或者XSRF，攻击者可以以用户的身份完成操作达到各种目的。

|  |  |
| --- | --- |
| 测试用例名称 | CSRF攻击 |
| 测试目的 | 防止CSRF攻击 |
| 测试条件 | 无 |
| 执行步骤 | 一：   * 检查应用系统中的敏感操作（如重启应用系统，添加管理员）中，是否有附加的CSRF-token。 * 检查提交表单是否有隐藏域的防止csrf的token字串； * 检查系统是否有全局的防止CSRF机制；   二：  去掉敏感操作中的附加token。 |
| 预期结果 | 应用具备防CSRF机制。  去掉token后表单提交失败，应用系统操作失败。 |
| 测试工具 | 无 |

# 七、Sql注入

SQL注入就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。

SQL注入通常分为Bool盲注，现错注入,Union注入。注入形式有可能是post,Get请求，也有可能是cookie,header等。由于目前针对SQL注入有非常多的工具可以检测，所以一般我们推荐的方法是，先用爬虫工具扫描一下，把可疑的URL先过滤出来，然后放在sqlmap工具里去执行，来最终确认是否存在问题。

当然也可以通过手工的方法来判断是否有注入，比较依赖个人的技术能力。例如通过修改动态参数，在参数值后加上 And 1=1和And 1=2或者’ aNd ‘a’=’a与’ aNd ‘a’=’b，提交后对返回内容作对比，如And 1=1页面返回内容和原始页面相同，而And 1=2时返回的页面不同，证明存在注入点。

详细的可参考阅读：

<http://baike.baidu.com/link?url=AvLpqKYA7hedgy6At98H0jVRzc7Ggv1iqJLrTO1fdv6vQKZEs8O9imZjCPm1MnLLZmAQa3PM030jqSUwEJ3g9q>

|  |  |
| --- | --- |
| 测试用例名称 | 防Sql注入 |
| 测试目的 | 防sql注入 |
| 测试条件 | 无 |
| 执行步骤 | 1. 在表单输入框中输入常见sql注入语句：And 1=1和And 1=2或者’ aNd ‘a’=’a与’ aNd ‘a’=’b 2. 使用扫描工具扫出存在sql注入的页面 3. 将上步扫出来的URL放到sqlmap工具里去执行 |
| 预期结果 | 1. 假如And 1=1页面返回内容和原始页面相同，而And 1=2时返回的页面不同，证明存在注入点。 2. 扫描工具未扫出sql注入页面、sqlmap未报告sql注入 |
| 测试工具 | AppScan、AWVS、明鉴Web应用弱点扫描器、sqlmap |

下面主要介绍下常用的检测工具

**NetSparker,APPSCAN + SQLMAP**

扫描工具（NetSparker,APPSCAN），在这里就不多介绍，这里主要介绍下SQLMAP。

sqlmap支持五种不同的注入模式：

* 基于布尔的盲注，即可以根据返回页面判断条件真假的注入
* 基于时间的盲注，即不能根据页面返回内容判断任何信息，用条件语句查看时间延迟语句是否执行（即页面返回时间是否增加）来判断
* 基于报错注入，即页面会返回错误信息，或者把注入的语句的结果直接返回在页面中
* 联合查询注入，可以使用union的情况下的注入
* 堆查询注入，可以同时执行多条语句的执行时的注入

sqlmap支持的数据库有

MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase和SAP MaxDB

**常用的使用选项：**

–cookie : 设置我们的cookie值

-u : 指定目标URL

-b : 获取DBMS banner

–current-db : 获取当前数据库

–current-user:获取当前用户

-r：sqlmap可以从一个文本文件中获取HTTP请求，这样就可以跳过设置一些其他参数（比如cookie，POST数据，等等）

--data ： 此参数是把数据以POST方式提交，sqlmap会像检测GET参数一样检测POST的参数。

--delay：可以设定两个HTTP(S)请求间的延迟，设定为0.5的时候是半秒，默认是没有延迟的。

--timeout：可以设定一个HTTP(S)请求超过多久判定为超时，10.5表示10.5秒，默认是30秒。

--level：共有五个等级，默认为1，sqlmap使用的payload可以在xml/payloads.xml中看到，你也可以根据相应的格式添加自己的payload。这个参数不仅影响使用哪些payload同时也会影响测试的注入点，GET和POST的数据都会测试，HTTP Cookie在level为2的时候就会测试，HTTP User-Agent/Referer头在level为3的时候就会测试。总之在你不确定哪个payload或者参数为注入点的时候，为了保证全面性，建议使用高的level值。

--risk ：共有四个风险等级，默认是1会测试大部分的测试语句，2会增加基于事件的测试语句，3会增加OR语句的SQL注入测试。有些时候，例如在UPDATE的语句中，注入一个OR的测试语句，可能导致更新的整个表，可能造成很大的风险。测试的语句同样可以在xml/payloads.xml中找到，你也可以自行添加payload。详细用法参考：sqlmap手册http://drops.wooyun.org/tips/143

# 八、文件上传

文件上传主要是对用户上传的文件类型判断不完善，导致攻击者上传非法类型的文件，从而对网站进行攻击。上传的文件可以是木马，病毒，恶意脚本或者WebShell等。

以WebShell为例，测试时将asp或php后门文件与网站服务器web目录下正常的网页文件混在一起，然后使用浏览器来访问这些后门，得到一个命令执行环境，以达到控制网站服务器的目的。（可以上传下载或者修改文件，操作数据库，执行任意命令等）。

具体测试的方法和手段，可以是在上传文件的过程中，通过后台截包的方式，把上传文件改为asp或jsp之类的脚本并提交执行（百度搜索一句话木马，或直接用中国菜刀、K8飞刀之类的黑客工具），以获得服务端访问权限。

|  |  |
| --- | --- |
| 测试用例名称 | 文件上传验证绕过一 |
| 测试目的 | 防止黑客上传恶意文件 |
| 测试条件 | 准备要上传的木马文件，修改文件后缀为系统允许的后缀 |
| 执行步骤 | 1. 上传伪造好的木马。 2. 使用webscarab或者burp抓包，将文件后缀改成原来的后缀名。 |
| 预期结果 | 服务端没有被绕过，不允许上传。 |
| 测试工具 | BurpSuit、webscarab |

|  |  |
| --- | --- |
| 测试用例名称 | 文件上传验证绕过二 |
| 测试目的 | 防止黑客上传恶意文件 |
| 测试条件 | 制作一句话木马，更改成客户端允许上传的后缀。 |
| 执行步骤 | 1. 上传伪造好的木马。 2. 使用webscarab或者burp抓包，修改Content-type的类型为服务端允许的类型。 |
| 预期结果 | 服务端没有被绕过，不允许上传。 |
| 测试工具 | BurpSuit、webscarab |

|  |  |
| --- | --- |
| 测试用例名称 | 文件上传验证绕过三 |
| 测试目的 | 防止黑客上传恶意文件 |
| 测试条件 | 制作一句话木马，更改成客户端允许上传的后缀。 |
| 执行步骤 | 1. 上传伪造好的木马。 2. 使用webscarab或者burp抓包，修改后缀为以下几类：  * 寻找黑马单之外的漏网之鱼（如asa，cer） * 大小写绕过（aSp，pHp） * 双后缀解析绕过（如：.jpg.jsp .jsp.jpg的后缀形式） |
| 预期结果 | 服务端没有被绕过，不允许上传。 |
| 测试工具 | BurpSuit、webscarab |

|  |  |
| --- | --- |
| 测试用例名称 | 上传文件检查 |
| 测试目的 | 防止黑客通过各种手段执行或获取到文件 |
| 测试条件 | 准备符合上传要求的文件 |
| 执行步骤 | 1. 上传后，文件名是否修改，文件名是否具备一定的随机性。 2. 上传后，文件所在路径是否可猜测。 3. 文件所在目录是否可执行。 |
| 预期结果 | 1. 文件名上传后被修改成随机文件名。 2. 文件存放的路径无法获知。 3. 文件所在目录不可执行。 |
| 测试工具 | 无 |

|  |  |
| --- | --- |
| 测试用例名称 | 任意文件读取、下载 |
| 测试目的 | 测试是否存在任意文件读取、下载 |
| 测试条件 | web应用有读取系统文件的权限。  （其他情况：如果web应用没有读取系统文件的权限，那么可以读取应用本身的文件，需要测试人员，自己跳转目录，拼接路径读取文件） |
| 执行步骤 | * Web应用运行的机器是windows   假设url路径为：index.do?file=/test/aa.jpg   1. index.do?file=/test/aa.jpg把/test/aa.jpg替换为../../../../../../../../../../../windows/win.ini  * Web应用运行的机器是linux   假设url路径为：download.jsp?file=/upload/1.pdf   1. download.jsp?file=/upload/1.pdf 把/upload/1.pdf 替换为./../../../../../../../../../../etc/passwd   （以上../不宜过少，应该尽量多。） |
| 预期结果 | 页面不能显示/etc/passwd文件内容，也不能下载到windows/win.ini文件，假如可以，那么证明漏洞存在。 |
| 测试工具 | 无 |

# 九、Web会话管理（session）

|  |  |
| --- | --- |
| 测试用例名称 | Web会话管理--会话自动过期 |
| 测试目的 | 会话自动过期 |
| 测试条件 | 有普通用户登录权限 |
| 执行步骤 | 1. 用户登录 2. 过10分钟后刷新当前页面（session过期时间根据实际需求） |
| 预期结果 | 用户已不在登录状态 |
| 测试工具 | 无 |

|  |  |
| --- | --- |
| 测试用例名称 | Web会话管理--HttpOnly和Secure属性设置 |
| 测试目的 | 防止cookie被窃取；cookie以安全方式传输 |
| 测试条件 | 有普通用户登录权限 |
| 执行步骤 | 1. 检查各cookie的HttpOnly属性 2. 检查各cookie的Secure属性 |
| 预期结果 | 1. HttpOnly的值为true 2. Secure的值为true |
| 测试工具 | 无 |

|  |  |
| --- | --- |
| 测试用例名称 | Web会话管理--防止会话劫持（建议项） |
| 测试目的 | 防止会话劫持 |
| 测试条件 | 有普通用户登录权限 |
| 执行步骤 | 1. 用户A在a机器登陆，获取cookie的值。 2. 在b机器使用以上获得的cookie值发送请求。 |
| 预期结果 | 请求被拒绝 |
| 测试工具 | Burpsuit、webscarab、 |

|  |  |
| --- | --- |
| 测试用例名称 | Web会话管理--限制重复登录（根据实际需求） |
| 测试目的 | 限制重复登录 |
| 测试条件 | 有普通用户登录权限 |
| 执行步骤 | 1. A用户登录机器a 2. A用户登录机器b |
| 预期结果 | a机器能告警提示或者自动退出并给出告警信息。 |
| 测试工具 | 无 |

|  |  |
| --- | --- |
| 测试用例名称 | Web会话管理（COOKIE） --防止会话固定 |
| 测试目的 | 防止会话固定 |
| 测试条件 | 有普通用户登录权限 |
| 执行步骤 | 1. 用户登录之后再次查看Cookie，是否够随机 |
| 预期结果 | 两次session不一致，够随机（不是用用户名、时间戳等生成的） |
| 测试工具 | 无 |

# 十、Web服务器信息

|  |  |
| --- | --- |
| 测试用例名称 | Web服务器信息—web服务器不允许公网访问 |
| 测试目的 | web服务器不允许公网访问 |
| 测试条件 | 无 |
| 执行步骤 | 从公网访问测试环境 |
| 预期结果 | 从公网不能访问测试环境 |
| 测试工具 | 无 |